嵌入式視覺

期末專題報告

智慧影像感測器

Image Super-Resolution Using Deep Convolutional Networks

基於深度卷積神經網路之影像超解析度

指導教授：陳慶瀚

學生：張翔珳

學號：104522065

E-mail：hwchang.miat@gmail.com

2017年1月

摘要

類神經網路為一種仿生物神經系統的數學模型，因為其非線性能力，以及對複雜資料學習適應能力強，在近年來被廣泛地應用在各種問題上，如函數逼近、分類、特徵擷取等等。在過去資訊科學家的研究下，類神經網路又產生各種不同的神經網路架構、階層數量，如卷積神經網路(Convolutional Neural Network, CNN)、徑向基底函數網路(Radial basis function network, RBFN)、遞歸神經網路(Recurrent neural network, RNN)等等，其中卷積神經網路的卷積層能夠取得空間域相鄰資料的關係，因而被應用在圖形處理上。

本研究是根據Chao[1]等人的研究論文SRCNN(Image Super-Resolution Using Deep Convolutional Networks)作進一步CNN於超解析度(Super-Resolution, SR)的介紹，並將其加入智慧影像感測器系統，以MIAT方法論進行IDEF0的階層式設計和控制器的Grafcet離散事件建模。

1. **問題描述**

由於人們對影像需求漸增，攝影機被應用在監控、消費性電子等各種領域，隨著影像高解析度的追求，也反映在成本上的提升，對於需大量部屬的監控攝影機來說，一般皆採用較低解析度影像，在監控廣度上也需要增加廣角範圍的拍攝，而時變的環境光源也增加監控的難易度，因此，設計一套系統能夠結合適應廣角變形的校正、超解析度影像與動態範圍調整器，便有其必要性。

廣角鏡頭讓攝影機得以容納更寬廣範圍的影像，但同時也造成影像廣角變形，傳統廣角變形校正是利用多項式對變形曲線進行擬合，一旦攝影機製程上的工差過大，造成平面鏡片上高度的非線性和不規則變形，以及鏡片座和晶片偏移所造成的離心誤差，則須同時套用多個校正方程式以改善誤差，因此採用類神經網路找出變形和校正影像座標空間的映射關係。

超解析度是一種能夠以較低成本的低解析度攝影機達到更佳影像品質的方法，一般超解析度可以分成兩種方式，一種是對單張影像進行影像內容的評估，在不同的影像內容進行插補，另一種則是以多張低解析度影像結合出高解析度影像，而本研究是採用SRCNN來實作超解析度，讓類神經網路學習低解析度和高解析度影像之間的映射關係。

動態範圍調整彌補了攝影機對於環境光源的對比強度，能夠改善偏暗或偏亮的影像品質。最簡單的做法是直方圖均勻化，以影像的灰階直方圖進行灰階最小和灰階最大到0和255的擴展，但影像品質將呈現過度強化或是造成灰階亮度多樣性的降低，於是便有BBHE(Brightness Preserving Bi-Histogram Equalization)以平均值做為分割子直方圖的門檻、BPDHE(Brightness Preserving dynamic histogram equalization)以直方圖的區域最大值進行個別的直方圖均勻化、DSIHE(Dualistic sub image histogram equalization)以直方圖平均和中值切割出子直方圖進行個別直方圖均勻化等改良演算法，而本研究採用兩種的結合，一種是自適應雙門檻直方圖均勻化ADPHE(Adaptive double plateau histogram equalization)[2]，對直方圖進行上下界的門檻化，另一種則是直方圖取代SHE(Simple histogram equalization)[3]，進而解決最低值和最高值灰階數量過高。

藉由這三種影像前處理，解決廣角變形、低解析度、自適應環境光源，進而達到影像品質的提升、成本降低。

1. **原理和方法**

一般CNN的架構可分為輸入層、卷積層、激活函數、池化層、全連接層、輸出層。而Chao[1]同時使用CNN和稀疏編碼特性，對相同大小的輸入影像和輸出影像實作非線性映射，進而達到影像超解析度。

稀疏編碼是參考生物神經網路的對於外界輸入刺激，僅部分5%~15%的神經元被激活的特性，讓類神經網路在訓練時僅以部分神經元去針對輸入和輸出的權重調整，能夠更有效率和精準地達到映射關係的對應，示意圖如圖1所示。該特性實作於類神經網路即是用ReLU(Rectified linear units)函數，如max(0, out)，輸出小於0的神經元便不影響和輸出至下一層神經元，達到部分激活、部分不激活的特性。

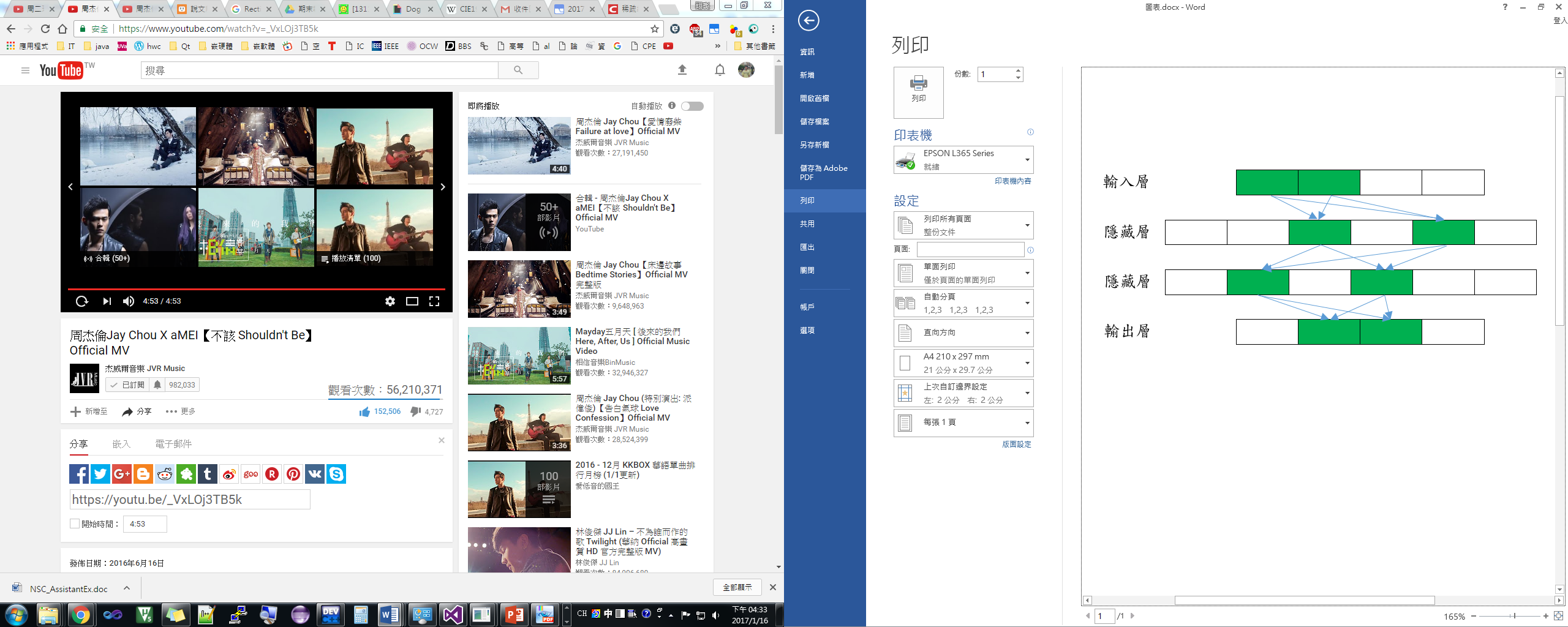


圖1 ReLU激活函數

SRCNN演算法步驟意義如下：

1. 對輸入影像作Bicubic雙立方插補成欲放大倍率的暫存影像。
2. 對暫存影像作特徵抽取，找出影像當中邊緣、非邊緣、紋理等特徵的高維度特徵影像，特徵擷取的部分是以學習得到的遮罩權重，對暫存影像作遮罩運算。
3. 基於稀疏編碼的特性，對高維度特徵影像作非線性映射，僅保留該像素點在高維度上的重要特徵。
4. 最後則是以近似於平均權重的遮罩對重要特徵作總合。

2~4步驟如圖2所示，該圖引用自Chao論文[1]。

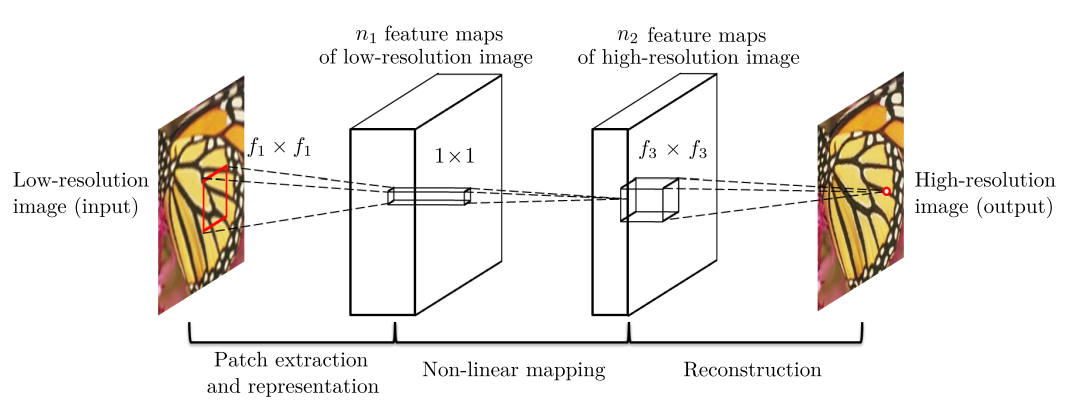


圖2 SRCNN演算步驟

其詳細演算法可分為學習和回想階段，步驟如下：

**學習階段**

目標影像：

將N張訓練影像Y切割成32\*32大小的等分區塊Z’

輸出影像：

1. 將N張訓練影像Y縮小成輸入影像X
2. 將輸入影像X透過Bicubic插補法放大成特定倍率影像Y’
3. 將所有影像Y’切割成32\*32大小的等分區塊
4. 對區塊作64次9\*9遮罩運算，資料維度將變成24\*24\*64
5. 對24\*24\*64作64\*32次5\*5遮罩運算成24\*24\*32影像（非線性映射）
6. 對24\*24\*32作32次5\*5遮罩運算，累加相同座標像素成20\*20影像Z

適應函數：

以Z和Z’的中心20\*20影像作均方差(Mean Square Error, MSE)計算

學習方法：

隨機選取區塊影像進行倒傳遞差值權重更新

**回想階段**

和學習階段的輸出影像計算過程相似，遮罩運算時複製邊界作遮罩元素填補，如下所示：

1. 將輸入影像A透過Bicubic插補法放大成特定倍率影像B
2. 將影像B作64次9\*9遮罩運算，資料維度將變成nr\*nc\*64
3. 對nr\*nc\*64作64\*32次5\*5遮罩運算成nr\*nc\*32影像（非線性映射）
4. 對nr\*nc\*32作32次5\*5遮罩運算，累加相同座標像素成nr\*nc輸出影像C
5. **系統架構**

使用MIAT方法論以IDEF0設計階層式系統架構分割，下層功能模組以Grafcet畫出細部運算流程，完成各功能模組之間的整合，目前已完成A43超解析度與A44動態範圍調整的軟體合成。

A0為智慧影像感測器系統的最上層模組，如圖所示，其輸入部分為從攝影機取得的Raw Image，以及從PC(Personal Computer)端載入的類神經元權重Neural weights，輸出部分，則是對攝影機的初始化(I2C)，以及訓練階段所需的變形影像(Distortion Image)和處理過後的影像(Processed Image)。

A0又分為五個子功能模組，如圖3所示，A1為控制設定器負責初始化A4和A5功能模組，A2部分則是負責實作DCMI傳輸和攝影機初始化的取像控制器，A2在取像後會輸出未經處理的Bayer pattern影像，A3模組則是負責Bayer pattern轉換到彩色像素、灰階像素，以及輸出當前像素座標點，A4模組為整體系統的核心影像增強器，負責三個功能的實現，分別是廣角變形校正、超解析度、動態範圍調整，輸出則為調整過後的像素，A5為影像傳輸器，負責將處理過後的影像傳輸至MCU(Microcontroller Unit)端或VGA monitor。



圖3 智慧影像感測IDEF0系統架構

A3功能模組是色彩轉換器，如圖4所示，又分為兩個子模組，A31負責將Bayer pattern Image轉換至RGB像素，A32模組則是將RGB像素再進一步轉換為灰階像素，轉換過程中會不斷累加座標並輸出給下一級模組。



圖4 色彩轉換模組

A4功能模組為影像增強器，如圖5所示，A4又分為A41到A44子功能模組，其中Init Parameters會初始化A41、A43和A44功能模組，A41為廣角變形校正器，計算變形座標和校正座標的映射關係，並以Bicubic插補器插補至欲放大倍率的影像，A42會將座標點和對應像素存入暫存的line buffer，以供應資料給後續的超解析度插補器A43，A43為SRCNN超解析度插補器，對輸入影像作一連串的特徵擷取、稀疏非線性映射篩選重要特徵、特徵加總，最後輸出插補像素，A44為動態範圍調整器，此功能模組在一開始有效灰階進入A4模組後，便開始統計灰階直方圖，根據上下界門檻和灰階取代建立灰階映射陣列，將插補後影像做直方圖均勻化。



圖5 影像增強功能模組

A5模組影像傳輸器，如圖6所示，各別以A51、A52的Line Buffer將影像透過DCMI和VGA控制器傳輸至MCU端和VGA monitor。



圖6 影像傳輸功能模組

A43功能模組的Grafcet控制器的部分如圖7到10所示，相關演算法步驟，如原理與方法所提到的內容，其軟體合成如附錄的程式碼。

|  |  |  |
| --- | --- | --- |
| 圖7 超解析度插補控制器 | 圖8 Patch extraction | |
| 圖9 Non-linear mapping | | 圖10 Reconstruction |

A44功能模組的Grafcet控制器如圖11所示，相關說明如表1所示。



圖11 A44功能模組Grafcet控制器

表1 A44功能模組說明

|  |  |  |
| --- | --- | --- |
| 狀態 | 動作 | 功能說明 |
| 440 | //Initial histogram bin to zero  i=j=0; for(int z=0;z < M;z++)bin[z]=0 | 直方圖和影像座標初始化 |
| 441 | //read image pixels  pixel = ima(i, j)  if(j == nc - 1) i++, j=0; else j++; | 取得影像對應座標像素  座標遞增 |
| 442 | //statistics histogram  bin[pixel]++ | 統計直方圖 |
| 443 | first\_one = last\_two = last\_one = -1;  bi=255; fi=0; head=0;  for(z = 0;z < WIN\_SIZE;z++)  　buf[i] = 0;  nz\_cnt=0;  max\_sum=0; max\_cnt=0; | 各項參數初始化：  SHE參數  ADPHE參數 |
| 444 | if(bin[bi]&&last\_one<0)  last\_one=bi;  elsif(bin[bi]&&last\_two<0) last\_two=bi;  bi--; | 找直方圖最後一個和倒數第二個非零的灰階索引值 |
| 445 | if(bin[fi] && first\_one < 0) first\_one=fi; | 找直方圖第一個  非零的灰階索引值 |
| 446 | //get local max sum  buf[head]=bin[fi];  max\_sum+=lmax(buf);  max\_cnt+=(buf[WIN\_SIZE\_2] >= max) ? 1 : 0;  head=(head+1)%WIN\_SIZE; | 計算直方圖的  區域最大值平均值 |
| 447 | //non-zero count  if(bin[fi++]) nz\_cnt++; | 統計直方圖出現次數  非零的數量 |
| 448 | //threshold calculation  Tup=max\_sum/(max\_cnt+1)  Tdown=min(nr \* nc, Tup\*nz\_cnt)/(M<<1)  index=1; N=0; i=0; j=0;  bin[first\_one]=0; bin[last\_one]=bin[last\_two]; | 計算  自適應上界、下界門檻值  取代第一個出現的灰階次數為0  將倒數第一個的數量取代為倒數第二個的數量 |
| 449 | //threshold histogram  z:=index; temp:=bin[z];  if(bin[z] >=Tup)  　temp:=Tup;  elsif(bin[z]==0) temp:=0;  elsif (bin[z]<=Tdown)  　temp:=Tdown; | 根據上界、下界  門檻化直方圖 |
| 4410 | //accumulative  bin[z]=temp + bin[z-1];  N+=temp;  index++; | 累加直方圖出現次數 |
| 4411 | //LUT output  bima[i][j]=bin[ima[i][j]] \* (M-1) / N;  if(j == nc - 1){ i++; j = 0;}  else j++; | 以輸入影像對  直方圖查表數值  至輸出影像 |
| 4412 | NULL | 上層模組轉移空狀態 |

1. **系統驗證**

A43功能模組結果圖如下表2所示，以SRCNN結果與Bicubic和APNN[4](Anisotropic Probability Neural Network)進行比較。

表2 實驗結果比較

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bicubic | APNN | | SRCNN | |
|  | C:\Users\MIAT\Desktop\MIAT\Master\APNN\APNN\20130806-1\bcb\testing_project\test.bmp | |  | |
| C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set5\BICUBIC.bmp | C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set5\test.bmp | | C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set5\SRCNN.bmp | |
| C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set14\BICUBIC.bmp | C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set14\test.bmp | | C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set14\SRCNN.bmp | |
| C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set5\BICUBIC.bmp | C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set5\test.bmp | | C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set5\SRCNN.bmp | |
| C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set14\BICUBIC.bmp | | C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set14\test.bmp | | C:\Users\MIAT\Desktop\MIAT\Master\reference\SRCNN_v1\SRCNN\Set14\SRCNN.bmp |

1. **結論**

以SRCNN作影像超解析度，優點在於能夠省去以人工選取對影像插補影響的特徵，讓類神經網路透過遮罩權重對輸出的影像品質優劣做學習，不斷地修正權重以達到最適宜的特徵選擇。SRCNN的插補效果遠優於其他插補方法，但相反地，其所耗費的計算資源和記憶體數量遠多於其他方法，所以將其設計成硬體做硬體加速，則有其必要性。

1. **參考資料**

[1] Chao et al. “Learning a Deep Convolutional Network for Image Super-Resolution”

[2] Liang et al. “A new adaptive contrast enhancement algorithm for infrared images based on double plateaus histogram equalization”

[3] Chang et al. “A simple histogram modification scheme for contrast enhancement”

[4] Chen et al. “Anisotropic Probabilistic Neural Network for Image Interpolation”

1. **附錄**

程式碼(僅附上A43超解析度功能模組的軟體合成，類神經網路權重在專案內)

|  |
| --- |
| #include <iostream>  #include <stdio.h>  #include "stdlib.h"  #include <string.h>  #include <time.h>  #include "bmp.h"  #include "array\_1d.h"  #include "array\_2d.h"  #define M 256  #define WIN\_SIZE 9  #define WIN\_SIZE\_2 (WIN\_SIZE >> 1)  #define max(x, y) (x > y ? x : y)  using namespace std;  typedef struct{  int \*size;  int num\_of\_size;  double \*weight\_conv;  }WEIGHT\_CONV;  void readWeight(char \*filename, WEIGHT\_CONV \*weight);  void readWeight\_reverse(char \*filename, WEIGHT\_CONV \*weight);  int \*i1D\_Initialize(int nr);  int \*\*i2D\_Initialize(int nr, int nc);  int \*\*\*i3D\_Initialize(int nr, int nc, int depth);  double \*d1D\_Initialize(int nr);  double \*\*d2D\_Initialize(int nr, int nc);  double \*\*\*d3D\_Initialize(int nr, int nc, int depth);  void grafcet43();  void action43();  void grafcet432();  void action432();  void grafcet433();  void action433();  void grafcet434();  void action434();  int Init\_available = 1;  int X43 = 1;  int X430 = 1, X431, X432, X433, X434, X435;  int X4320 = 1, X4321, X4322, X4323, X4324, X4325;  int X4330 = 1, X4331, X4332, X4333, X4334, X4335;  int X4340 = 1, X4341, X4342, X4343, X4344, X4345;  unsigned char \*\*ima, \*\*bima;  int nr, nc;//image height  int i, j, k, l, m, n;  char filename[128];  bool isvalid;  clock\_t start, end;  //read weight  WEIGHT\_CONV weights\_conv1;  WEIGHT\_CONV weights\_conv2;  WEIGHT\_CONV weights\_conv3;  WEIGHT\_CONV biases\_conv1;  WEIGHT\_CONV biases\_conv2;  WEIGHT\_CONV biases\_conv3;  int depth, depth2, depth3;  double \*\*\*conv1\_data, \*\*\*conv2\_data, \*\*conv3\_data;  double \*\*conv1\_subfilter, \*\*conv2\_subfilter, \*\*conv3\_subfilter;  int conv1\_patchsize, conv1\_patchsize\_2;  int conv2\_patchsize, conv2\_patchsize\_2;  int conv3\_patchsize, conv3\_patchsize\_2;  void grafcet43(){  if(X430 == 1 && Init\_available == 1){ X430 = 0; X431 = 1;}  else if(X431 == 1 && X43 == 1){ X431 = 0; X432 = 1;}  else if(X432 == 1 && X4325 == 1){ X432 = 0; X4320 = 1; X4325 = 0; X433 = 1;}  else if(X433 == 1 && X4335 == 1){ X433 = 0; X4330 = 1; X4335 = 0; X434 = 1;}  else if(X434 == 1 && X4345 == 1){ X434 = 0; X4340 = 1; X4345 = 0; X435 = 1;}  else if(X435 == 1){ X435 = 0; X430 = 1; exit(0);}  action43();  }  void action43(){  if(X431 == 1){  isvalid = Read\_BMP("lenna\_bicubic.bmp", ima, nr, nc);  if (!isvalid) exit(1);    readWeight("weights\_conv1.txt", &weights\_conv1);  readWeight("weights\_conv2.txt", &weights\_conv2);  readWeight\_reverse("weights\_conv3.txt", &weights\_conv3);    readWeight("biases\_conv1.txt", &biases\_conv1);  readWeight("biases\_conv2.txt", &biases\_conv2);  readWeight("biases\_conv3.txt", &biases\_conv3);    //convolution operation  //81\*64 => 9\*9\*64  depth = weights\_conv1.size[1];  conv1\_data = d3D\_Initialize(nr, nc, depth);//nr \* nc \* 64  conv1\_patchsize = sqrt(weights\_conv1.size[0]); //9  conv1\_patchsize\_2 = conv1\_patchsize >> 1; //4    conv1\_subfilter = d2D\_Initialize(conv1\_patchsize, conv1\_patchsize);    //conv 2 layer  //64\*25\*32  depth2 = weights\_conv2.size[2];  conv2\_data = d3D\_Initialize(nr, nc, weights\_conv2.size[2]);  conv2\_patchsize = sqrt(weights\_conv2.size[1]);  conv2\_patchsize\_2 = conv2\_patchsize >> 1;    conv2\_subfilter = d2D\_Initialize(conv2\_patchsize, conv2\_patchsize);    //conv 3 layer  //32 \* 25  depth3 = weights\_conv3.size[0];  conv3\_data = d2D\_Initialize(nr, nc);  conv3\_patchsize = sqrt(weights\_conv3.size[1]);  conv3\_patchsize\_2 = conv3\_patchsize >> 1;    conv3\_subfilter = d2D\_Initialize(conv3\_patchsize, conv3\_patchsize);    //start of image processing  start = clock();    }else if(X432 == 1){  grafcet432();  }else if(X433 == 1){  grafcet433();  }else if(X434 == 1){  grafcet434();  }else if(X435 == 1){  }  }  void grafcet432(){  if(X4320 == 1 && X432 == 1){ X4320 = 0; X4321 = 1;}  else if(X4321 == 1){ X4321 = 0; X4322 = 1;}  else if(X4322 == 1){ X4322 = 0; X4323 = 1;}  else if(X4323 == 1){ X4323 = 0; X4324 = 1;}  else if(X4324 == 1 && k < depth){ X4324 = 0; X4322 = 1;}  else if(X4324 == 1 && !(k < depth)){ X4324 = 0; X4325 = 1;}  else if(X4325 == 1){ X4325 = 0; X4320 = 1;}  action432();  }  void action432(){  if(X4321 == 1){ k = 0;}  else if(X4322 == 1){  //reshape 81 => 9\*9  for(l = 0;l < conv1\_patchsize;l++){  for(m = 0;m < conv1\_patchsize;m++){  int index = l \* conv1\_patchsize +  m + k \* weights\_conv1.size[0];  double weight = weights\_conv1.weight\_conv[index];  conv1\_subfilter[l][m] = weight;  }  }  }else if(X4323 == 1){  for(i = 0;i < nr;i++){  for(j = 0;j < nc;j++){  double sum\_of\_conv1 = 0.0f;  for(l = - conv1\_patchsize\_2;l <= conv1\_patchsize\_2;l++){  for(m = - conv1\_patchsize\_2;m <= conv1\_patchsize\_2;m++){  int y = i + l;  int x = j + m;  x = x >= 0 ? (x < nc ? x : nc - 1) : 0;  y = y >= 0 ? (y < nr ? y : nr - 1) : 0;  sum\_of\_conv1 += (double)ima[y][x] \* conv1\_subfilter[l + conv1\_patchsize\_2][m + conv1\_patchsize\_2];  }  }  sum\_of\_conv1 = sum\_of\_conv1 / 255.0f + biases\_conv1.weight\_conv[k];  conv1\_data[i][j][k] = max(sum\_of\_conv1, 0.0f);  }  }  }else if(X4324 == 1){  k++;  }else if(X4325 == 1){  }  }  void grafcet433(){  if(X4330 == 1 && X433 == 1){ X4330 = 0; X4331 = 1;}  else if(X4331 == 1){ X4331 = 0; X4332 = 1;}  else if(X4332 == 1){ X4332 = 0; X4333 = 1;}  else if(X4333 == 1 && n < depth){ X4333 = 0; X4332 = 1;}  else if(X4333 == 1 && !(n < depth)){ X4333 = 0; X4334 = 1;}  else if(X4334 == 1 && k < depth2){ X4334 = 0; X4332 = 1;}  else if(X4334 == 1 && !(k < depth2)){ X4334 = 0; X4335 = 1;}  else if(X4335 == 1){ X4335 = 0; X4330 = 1;}  action433();  }  void action433(){  if(X4331 == 1){ k = 0; n = 0;}  else if(X4332 == 1){  //reshape 25 => 5\*5  for(l = 0;l < conv2\_patchsize;l++){  for(m = 0;m < conv2\_patchsize;m++){  int index = (l \* conv2\_patchsize + m + n \* weights\_conv2.size[1]) \* weights\_conv2.size[2] + k;  conv2\_subfilter[m][l] = weights\_conv2.weight\_conv[index];  }  }  }else if(X4333 == 1){  //apply conv2\_subfilter on conv1\_data  for(i = 0;i < nr;i++){  for(j = 0;j < nc;j++){  double sum\_of\_conv2 = 0.0f;  for(l = - conv2\_patchsize\_2;l <= conv2\_patchsize\_2;l++){  for(m = - conv2\_patchsize\_2;m <= conv2\_patchsize\_2;m++){  int y = i + l;  int x = j + m;  x = x >= 0 ? (x < nc ? x : nc - 1) : 0;  y = y >= 0 ? (y < nr ? y : nr - 1) : 0;  sum\_of\_conv2 += conv2\_subfilter[l + conv2\_patchsize\_2][m + conv2\_patchsize\_2] \*  conv1\_data[y][x][n];  }  }  conv2\_data[i][j][k] += sum\_of\_conv2;  }  }  n++;  }else if(X4334 == 1){  for(i = 0;i < nr;i++){  for(j = 0;j < nc;j++){  conv2\_data[i][j][k] = max(conv2\_data[i][j][k] + biases\_conv2.weight\_conv[k], 0.0f);  }  }  printf("doing convolution on layer 2... %d\n", k);  k++; n=0;  }else if(X4335 == 1){  }  }  void grafcet434(){  if(X4340 == 1 && X434 == 1){ X4340 = 0; X4341 = 1;}  else if(X4341 == 1){ X4341 = 0; X4342 = 1;}  else if(X4342 == 1){ X4342 = 0; X4343 = 1;}  else if(X4343 == 1){ X4343 = 0; X4344 = 1;}  else if(X4344 == 1 && k < depth3){ X4344 = 0; X4342 = 1;}  else if(X4344 == 1 && !(k < depth3)){ X4344 = 0; X4345 = 1;}  else if(X4345 == 1){ X4345 = 0; X4340 = 1;}  action434();  }  void action434(){  if(X4341 == 1){ k = 0;}  else if(X4342 == 1){  //reshape 25 => 5\*5  for(l = 0;l < conv3\_patchsize;l++){  for(m = 0;m < conv3\_patchsize;m++){  int index = l \* conv3\_patchsize + m + k \* weights\_conv3.size[1];  conv3\_subfilter[m][l] = weights\_conv3.weight\_conv[index];  }  }  }else if(X4343 == 1){  for(i = 0;i < nr;i++){  for(j = 0;j < nc;j++){  double sum\_of\_conv3 = 0.0f;  for(l = - conv3\_patchsize\_2;l <= conv3\_patchsize\_2;l++){  for(m = - conv3\_patchsize\_2;m <= conv3\_patchsize\_2;m++){  int y = i + l;  int x = j + m;  x = x >= 0 ? (x < nc ? x : nc - 1) : 0;  y = y >= 0 ? (y < nr ? y : nr - 1) : 0;  sum\_of\_conv3 += conv3\_subfilter[l + conv3\_patchsize\_2][m + conv3\_patchsize\_2]  \* conv2\_data[y][x][k];  }  }  conv3\_data[i][j] = conv3\_data[i][j] + sum\_of\_conv3;  }  }  }else if(X4344 == 1){  k++;  }else if(X4345 == 1){  for(i = 0;i < nr;i++){  for(j = 0;j < nc;j++){  conv3\_data[i][j] += biases\_conv3.weight\_conv[0];  }  }    bima = UC2D(nr, nc);  for(i = 0;i < nr;i++){  for(j = 0;j < nc;j++){  bima[i][j] = (unsigned char)(conv3\_data[i][j] \* 255.0f);  }  }  Write\_BMP\_8bits("bima.bmp", bima, nr, nc);  //end of image processing    end = clock();    printf("execution time(s) = %lf\n", (double)(end - start) / CLOCKS\_PER\_SEC);  }  }  int main(int argc, char\*\* argv) {    while(1){  grafcet43();  }    return 1;  }  void readWeight\_reverse(char \*filename, WEIGHT\_CONV \*weight){    FILE \*fptr;  const int BUFFER\_SIZE = 80;  char header[BUFFER\_SIZE];  char \*hptr = header;  int i, j, k, l, m;    fptr = fopen(filename, "r");    if(fptr == NULL){  printf("Fail to open file\n");  return;  }    fgets(hptr, BUFFER\_SIZE, fptr);    //get number of dimension  weight->num\_of\_size = 0;  for(i = 0;hptr[i] != NULL;i++){  if(hptr[i] == ' '){  weight->num\_of\_size++;  }  }    weight->size = (int\*)malloc(sizeof(int) \* weight->num\_of\_size);    if(weight->size == NULL){  printf("Fail to allocate memory\n");  return;  }    int total\_size = 1;  char \*token = strtok(hptr, " ");  for(i = 0;i < weight->num\_of\_size;i++){    weight->size[i] = atoi(token);  total\_size \*= weight->size[i];  printf("[%d] = %d\t", i, weight->size[i]);    token = strtok(NULL, " ");  }    printf("\n");  //high-dimension first  weight->weight\_conv = d1D\_Initialize(total\_size);    if(weight->weight\_conv == NULL){  printf("Fail to allocate memory\n");  return;  }    if(weight->num\_of\_size == 3){  //not implement  }else if(weight->num\_of\_size == 2){  //weight[0][0] -> weight[1][0] -> weight[2][0]  int patchsize = sqrt(weight->size[1]);  for(k = 0;k < patchsize;k++){  for(l = 0;l < patchsize;l++){  for(j = 0;j < weight->size[0];j++){  double tmp;  fscanf(fptr, "%lf", &tmp);  weight->weight\_conv[k \* patchsize + l + j \* weight->size[1]] = tmp;  //printf("weight\_conv[%d, %d] = %lf\n", j, k \* patchsize + l, tmp);  }  }  }  /\*for(k = 0;k < patchsize;k++){  for(l = 0;l < patchsize;l++){  printf("weight\_conv[%d, %d] = %lf\n", k, l, weight->weight\_conv[k \* patchsize + l + 0 \* weight->size[1]]);  }  }\*/  }else if(weight->num\_of\_size == 1){  //not implement  }  fclose(fptr);  }  void readWeight(char \*filename, WEIGHT\_CONV \*weight){    FILE \*fptr;  const int BUFFER\_SIZE = 80;  char header[BUFFER\_SIZE];  char \*hptr = header;  int i, j, k, l, m;    fptr = fopen(filename, "r");    if(fptr == NULL){  printf("Fail to open file\n");  return;  }    fgets(hptr, BUFFER\_SIZE, fptr);    //get number of dimension  weight->num\_of\_size = 0;  for(i = 0;hptr[i] != NULL;i++){  if(hptr[i] == ' '){  weight->num\_of\_size++;  }  }    weight->size = (int\*)malloc(sizeof(int) \* weight->num\_of\_size);    if(weight->size == NULL){  printf("Fail to allocate memory\n");  return;  }    int total\_size = 1;  char \*token = strtok(hptr, " ");  for(i = 0;i < weight->num\_of\_size;i++){    weight->size[i] = atoi(token);  total\_size \*= weight->size[i];  printf("[%d] = %d\t", i, weight->size[i]);    token = strtok(NULL, " ");  }    printf("\n");  //high-dimension first  weight->weight\_conv = d1D\_Initialize(total\_size);    if(weight->weight\_conv == NULL){  printf("Fail to allocate memory\n");  return;  }    if(weight->num\_of\_size == 3){  //64\*25\*32  //data[0][0][0] -> data[1][0][0] -> ... -> data[0][1][0]  int patchsize = sqrt(weight->size[1]);  for(i = 0;i < weight->size[2];i++){//32  for(j = 0;j < weight->size[1];j++){//25  for(k = 0;k < weight->size[0];k++){//64  double tmp;  fscanf(fptr, "%lf", &tmp);  weight->weight\_conv[(j + k \* weight->size[1]) \* weight->size[2] + i] = tmp;  /\*if((k == 0 && j == 0 && i == 1) ||  (k == 1 && j == 0 && i == 0) ||  (k == 0 && j == 1 && i == 0))  printf("weight\_conv[%d, %d, %d] = %lf\n", k, j, i, tmp);\*/  }  }  }  }else if(weight->num\_of\_size == 2 && weight->size[1] == 1){  int patchsize = weight->size[0];  for(k = 0;k < patchsize;k++){  double tmp;  fscanf(fptr, "%lf", &tmp);  weight->weight\_conv[k] = tmp;  //printf("weight\_conv[%d, %d] = %lf\n", k, j, tmp);  }  }else if(weight->num\_of\_size == 2){  int patchsize = sqrt(weight->size[0]);  for(j = 0;j < weight->size[1];j++){  for(k = 0;k < patchsize;k++){  for(l = 0;l < patchsize;l++){  double tmp;  fscanf(fptr, "%lf", &tmp);  weight->weight\_conv[l \* patchsize + k + j \* weight->size[0]] = tmp;  //printf("weight\_conv[%d, %d] = %lf\n", k, j, tmp);  }  }  }  }else if(weight->num\_of\_size == 1){  for(k = 0;k < weight->size[0];k++){  double tmp;  fscanf(fptr, "%lf", &tmp);  weight->weight\_conv[k] = tmp;  //printf("weight\_conv[%d] = %lf\n", k, tmp);  }  }  fclose(fptr);  }  int \*i1D\_Initialize(int nr){  if(nr <= 0) return NULL;    int \*src = (int\*)calloc(nr, sizeof(int));    if(src == NULL){  printf("Fail to allocate memory\n");  return NULL;  }    return src;  }  int \*\*i2D\_Initialize(int nr, int nc){  int i;    if(nr <= 0 || nc <= 0) return NULL;    int \*\*src = (int\*\*)malloc(sizeof(int\*) \* nr);    if(src == NULL){  printf("Fail to allocate memory\n");  return NULL;  }    for(i = 0;i < nr;i++){  src[i] = i1D\_Initialize(nc);  }  return src;  }  int \*\*\*i3D\_Initialize(int nr, int nc, int depth){  int i;    if(nr <= 0 || nc <= 0 || depth <= 0) return NULL;    int \*\*\*src = (int\*\*\*)malloc(sizeof(int\*\*) \* nr);    if(src == NULL){  printf("Fail to allocate memory\n");  return NULL;  }    for(i = 0;i < nr;i++){  src[i] = i2D\_Initialize(nc, depth);  }  return src;  }  double \*d1D\_Initialize(int nr){  if(nr <= 0) return NULL;    double \*src = (double\*)calloc(nr, sizeof(double));    if(src == NULL){  printf("Fail to allocate memory\n");  return NULL;  }    return src;  }  double \*\*d2D\_Initialize(int nr, int nc){  int i;    if(nr <= 0 || nc <= 0) return NULL;    double \*\*src = (double\*\*)malloc(sizeof(double\*) \* nr);    if(src == NULL){  printf("Fail to allocate memory\n");  return NULL;  }    for(i = 0;i < nr;i++){  src[i] = d1D\_Initialize(nc);  }  return src;  }  double \*\*\*d3D\_Initialize(int nr, int nc, int depth){  int i;    if(nr <= 0 || nc <= 0 || depth <= 0) return NULL;    double \*\*\*src = (double\*\*\*)malloc(sizeof(double\*\*) \* nr);    if(src == NULL){  printf("Fail to allocate memory\n");  return NULL;  }    for(i = 0;i < nr;i++){  src[i] = d2D\_Initialize(nc, depth);  }  return src;  } |